

Mobiclip SDK for Nintendo CTR Encoding Guide

Version 1.0.11

**The content of this document is highly confidential
and should be handled accordingly.**

Confidential

These coded instructions, statements, and computer programs contain proprietary information of Nintendo and its licensed developers and are protected by national and international copyright laws. They may not be disclosed to third parties or copied or duplicated in any form, in whole or in part, without the prior written consent of Nintendo.

Table of Contents

1	Introduction	6
1.1	Video editing tools.....	6
1.2	Compatibilities issues	6
2	Video preparation.....	7
2.1	General recommendations.....	7
2.2	Cropping.....	10
2.3	De-interlace.....	14
2.4	Levels adjustments	16
2.5	Resize	19
3	Processing 3D resources.....	22
3.1	Case 1: One source video for each eye	25
3.2	Case 2: One video containing both left and right eyes	26
3.3	About pre-rotating video.....	28
3.4	About CTR liquid crystal display frequency and video frame rate	29
3.4.1	Changing movie frame rate value	29
3.4.2	Inserting video images to match the LCD refresh rate.....	31
4	Avisynth Filters Information Reference.....	32
4.1	AssumeFPS filter	32
4.2	ConvertFPS filter.....	32
5	Audio Encoding	33
6	Mobiclip Video for Windows(VFW) codec description	35
6.1	Access the VFW codec configuration menu	35
6.1.1	Selecting the Mobiclip VFW codec for NINTENDO CTR	35
6.1.2	Opening the Mobiclip VFW codec for NINTENDO CTR configuration dialog	36
6.1.3	Tuning the parameters of the Mobiclip VFW codec for NINTENDO CTR.....	36
6.2	VFW codec configuration reference	37
6.2.1	Encoding Type.....	38
6.2.2	Multipass encoding howto	45
6.3	VFW codec configuration walkthrough	47
6.3.1	Constant Quality encoding scheme	47
6.3.2	Multipass CBR encoding scheme	48
6.3.3	Multipass VBR encoding scheme	49
6.3.4	Single Pass CBR encoding scheme	50

7	Conversion to MoFlex file format.....	51
7.1	MoflexTool command line parameters description	52
7.2	MoflexTool command line examples	55
7.2.1	Display MoflexTool help on the command line screen.....	55
7.2.2	Display AVI input file properties on the command line screen.....	55
7.2.3	Display WAV input file properties on the command line screen	55
7.2.4	Display existing Moflex input file properties on the command line screen	55
7.2.5	Display a combination of AVI, WAV and Moflex input files properties on the command line screen	56
7.2.6	Retrieval of the first video track of an AVI file	56
7.2.7	Retrieval of the first video track of an AVI file with timeline generation	56
7.2.8	Conversion of the first audio track of an AVI file to a specific format.....	57
7.2.9	Use of an external audio track in WAV file.....	57
7.2.10	Use of multiple audio tracks in WAV files	58
7.2.11	Use of multiple audio tracks in WAV files and AVI files.....	58
7.2.12	Use of -video parameter options for a 3D interleaved video	59
7.2.13	Use of -video parameter options for a 3D interleaved pre-rotated video.....	59
7.3	MoflexTool file and stream options usage	60

Figures

Figure 1 VirtualDub video frame rate control dialog box	8
Figure 2 Opening Filters section in VirtualDub	10
Figure 3 Filters section opened in VirtualDub	11
Figure 4 Selection of null transform filter in VirtualDub	11
Figure 5 Tuning of cropping parameters in VirtualDub	12
Figure 6 Tuning of cropping parameters done in VirtualDub	13
Figure 7 Before de-interlace(on the left) and after de-interlace(on the right)	14
Figure 8 The various de-interlace techniques in VirtualDub, Yadif is selected	15
Figure 9 Levels adjustments in VirtualDub, Step 1	16
Figure 10 Levels adjustments in VirtualDub, Step 2	17
Figure 11 Levels adjustments in VirtualDub, Step 3	17
Figure 12 Levels adjustments in VirtualDub, Step 4	18
Figure 13 Resizing in VirtualDub, Step 1	20
Figure 14 Resizing in VirtualDub, Step 2	21
Figure 15 Exemple of a top to bottom layout	22
Figure 16 Example of a side by side layout	23
Figure 17 Avisynth script that interleaves two different source videos	25
Figure 18 Example of 3D source video with left and right eyes placed side by side	26
Figure 19 Avisynth script that interleaves left and right eye image from the same source video	26
Figure 20 Add audio track inside the Avisynth script	27
Figure 21 Example of pre-rotated video	28
Figure 22 Frame rate change to 29.9155 fps	30
Figure 23 Frame rate change to 59.831 fps of a 3D interleaved movie	30
Figure 24 Frame rate conversion to 29.9155 fps	31
Figure 25 Frame rate conversion to 29.9155 and interleaving of left/right eye image	31

1 Introduction

This document will give you step by step instructions to create videos in the Moflex file format.

1.1 Video editing tools

The free software VirtualDub and Avisynth tools will be used for the description of video preparation and encoding. We recommend using these tools to process and encode your videos.

You can also use “off the shelf” products from other software vendors.

VirtualDub is available for download at the following website:

<http://www.virtualdub.org/>

Avisynth is available for download at the following website:

http://avisynth.org/mediawiki/Main_Page

You're also invited to read the document *CTR-Mobiclip_SDK_Quick_Start_Guide-en_US.pdf* for an overview of the Mobiclip SDK structure and content.

1.2 Compatibilities issues

Some applications do not conform to the Video for Windows 1.1 standard (VFW 1.1) as they do not provide the codec with the frames per second information or the total number of frames information.

Whenever the codec detects such an incompatible application, it will display a popup that tells the user what information it is lacking.

2 Video preparation

This chapter describes the recommended procedures to prepare a video source before encoding it with the Mobiclip Video for Windows codec.

2.1 General recommendations

Video preparation and processing are very important. They will contribute to half of the final video quality, the other half coming from the video codec itself.

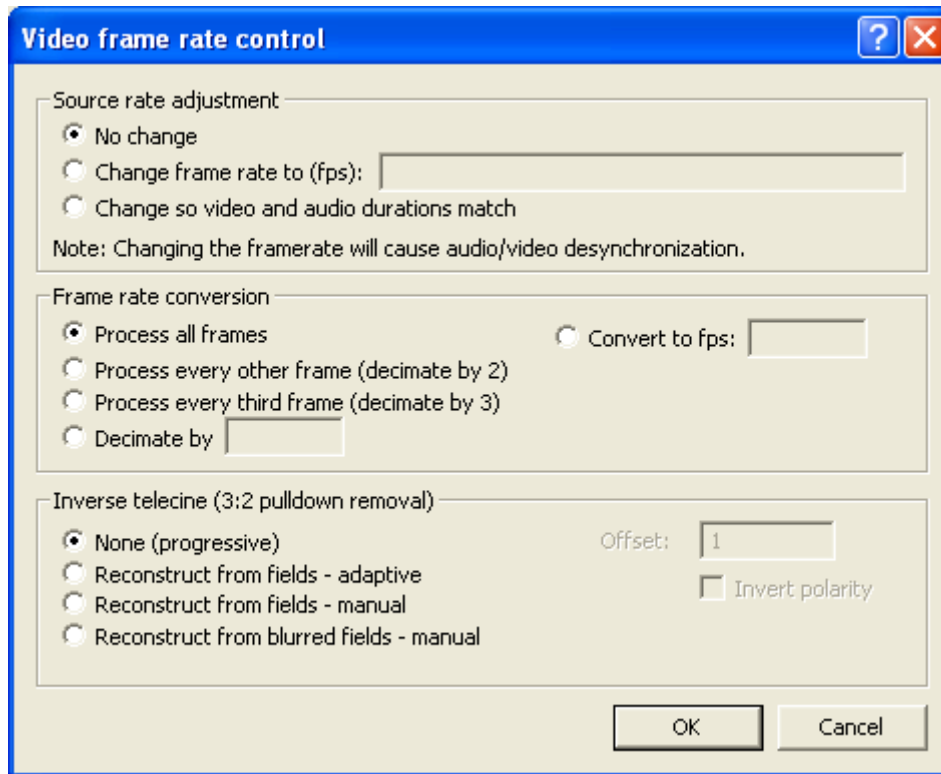
1. **Use the best source video you can get**
2. **Resolution of the source video should be big, at least the screen target resolution that is 400x240 for CTR ULCD screen and 320x240 for CTR lower screen**

Most processes (like contrast/gamma) will produce better results if applied to the original high-resolution video source, because they are applied to more pixels.
3. **Delay the final downsizing to one of the CTR target resolutions until as late as possible in your processing pipeline**
4. **Source video should be uncompressed or not compressed very much**

5. **When lowering frame rate, only use “Frame rate conversion” methods.**

See the **Video/Frame rate** menu in VirtualDub.

Figure 1 VirtualDub video frame rate control dialog box



Note: You should never use this method on a 3D interleaved video source.

6. **Do your best to minimize any changes in the aspect ratio of the source and final video (in general 10% difference is ok)**

Your source videos will probably have an aspect ratio of 4:3 (TV), 16:9 (widescreen cinema) or 2.35 (CinemaScope). The CTR ULCD screen aspect ratio is 5:3 (400x240) and the CTR lower screen aspect ratio is 4:3 (320x240). To convert from source aspect ratio to one of the CTR aspect ratios, you might have to crop your video (remove a part of the image) or add a letterbox.

7. **Downsize using a “Precise bicubic” (or better) method**

You can choose the target resolution of the videos, please be aware that the target resolution must be multiples of 16.

Again, do not forget to preserve the original aspect ratio.

8. When storing intermediate movies, use a lossless video codec

We recommend using Huffvuv for storing intermediate files.

The Huffvuv codec is available at the following website.

See <http://neuron2.net/www.math.berkeley.edu/benrg/huffvuv.html>

2.2 Cropping

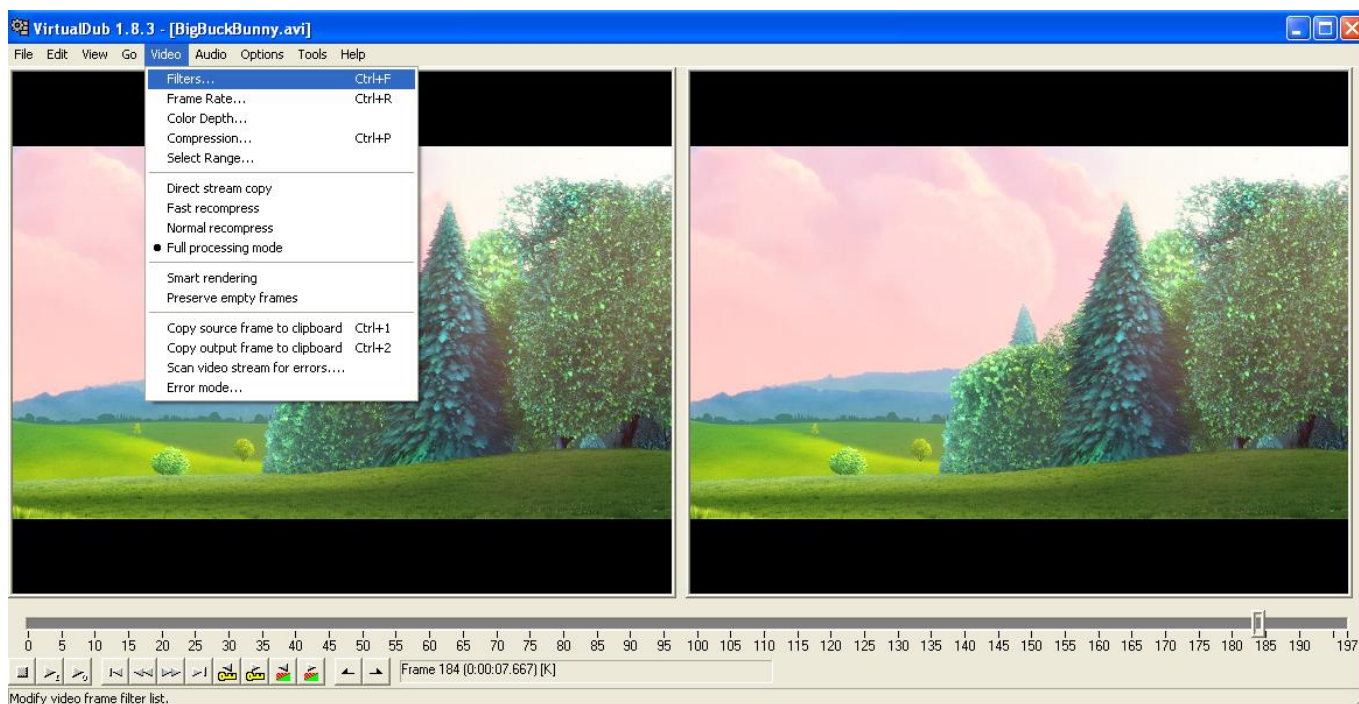
The objective is to delete letterboxing stripes or other unwanted zones within the original footage. We will select only the part of the picture that we want to encode.

Note: If you intend to encode all the video source content you can skip this step.

In order to crop, you need to select:

From the **Video** menu, click on **filter** to **add** the **null transform** and the **cropping** filter.

Figure 2 Opening Filters section in VirtualDub



Creative commons (c) copyright Blender Foundation | www.bigbuckbunny.org

Figure 3 Filters section opened in VirtualDub

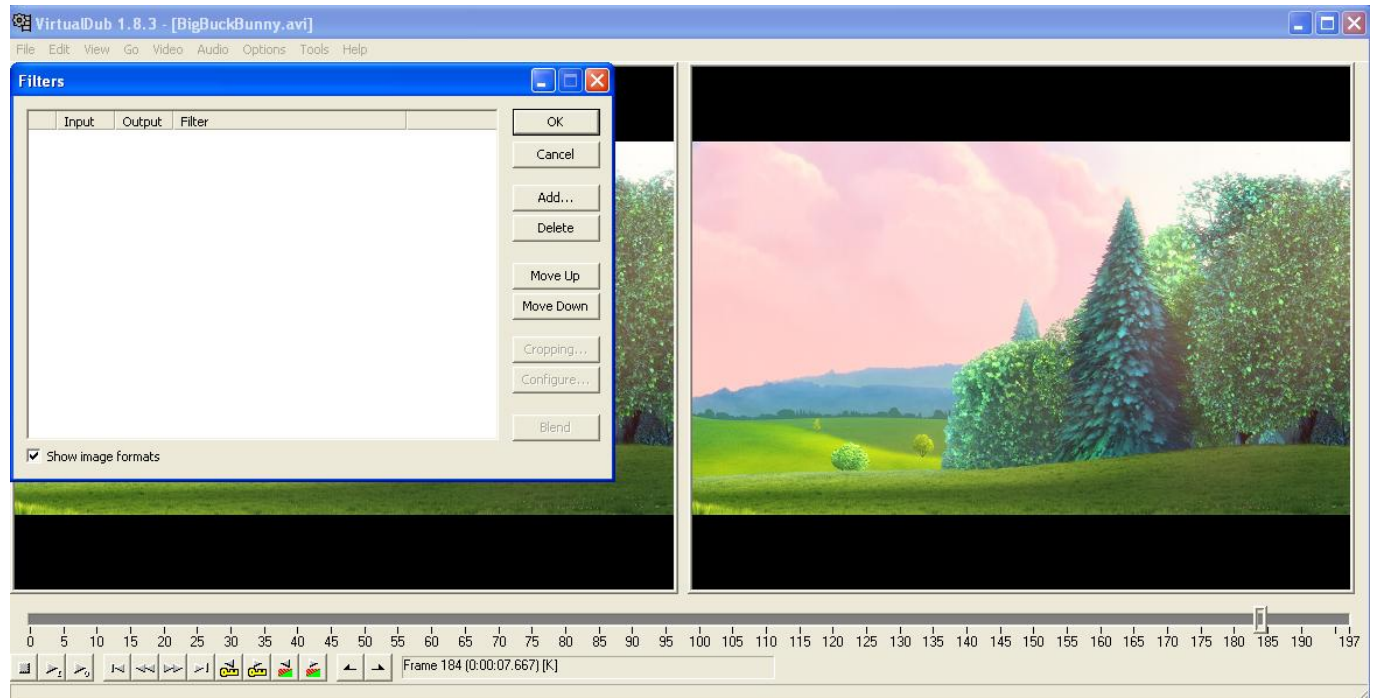
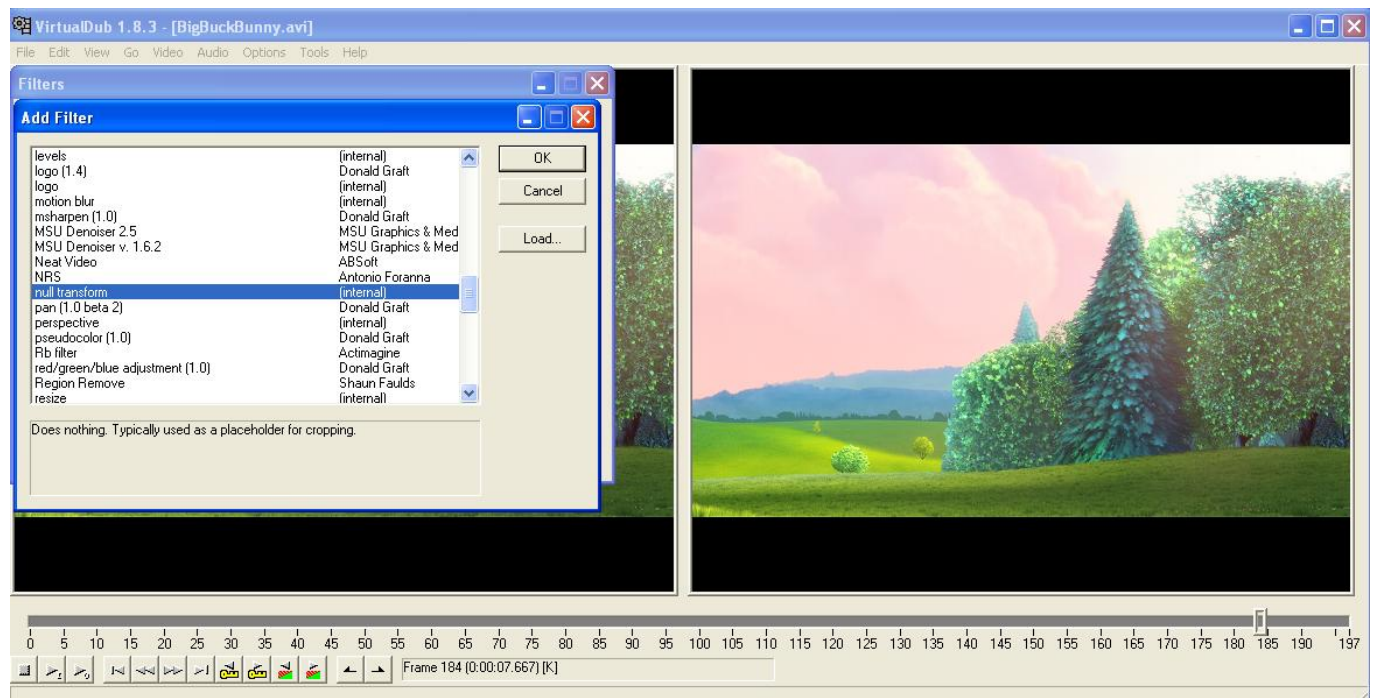
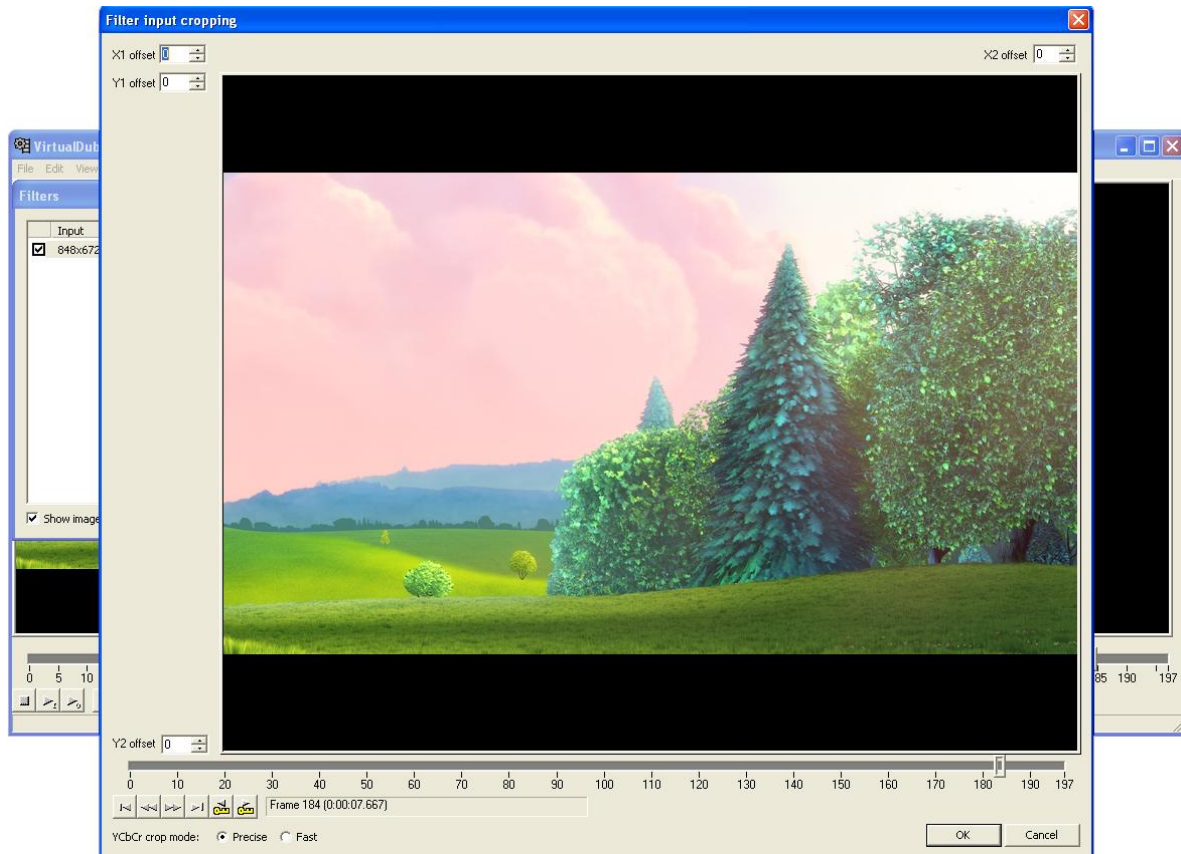


Figure 4 Selection of null transform filter in VirtualDub



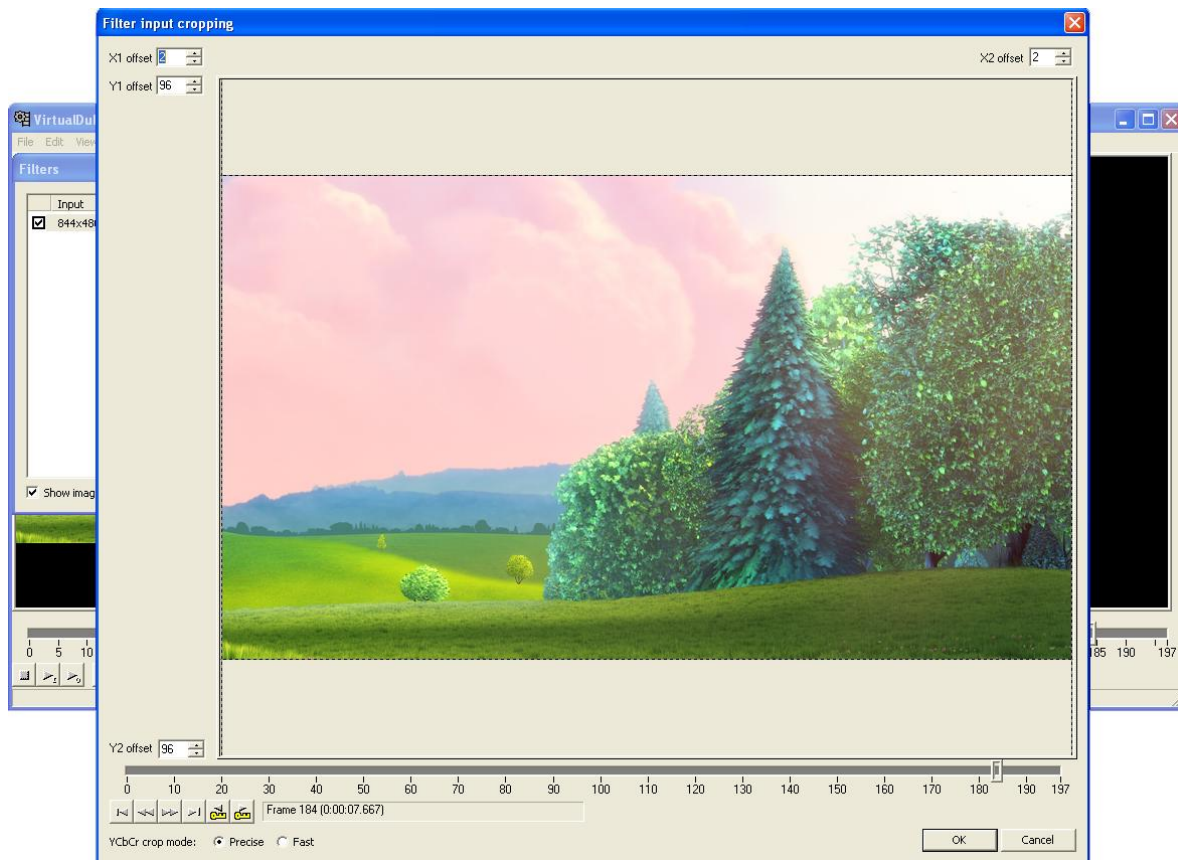
By setting the parameters **X1 offset**, **X2 offset**, **Y1 offset** and **Y2 offset** you can select the zone of the footage that you want to get rid of. In the case below, we want to remove the letterboxing black bars above and below the movie image.

Figure 5 Tuning of cropping parameters in VirtualDub



Once all cropping parameters have been set, you can click OK to close the crop window:

Figure 6 Tuning of cropping parameters done in VirtualDub



2.3 De-interlace

With some footage you can see some horizontal lines during scenes with fast movement.

This is because the video source has been captured in an interlaced fashion, like most camcorders do, or because the video source was prepared for television broadcast.

We need to remove these line artefacts in order to get clean pictures.

This process is called de-interlacing.

Note: If the video source is progressive, like most computer generated footages, you can skip this step.

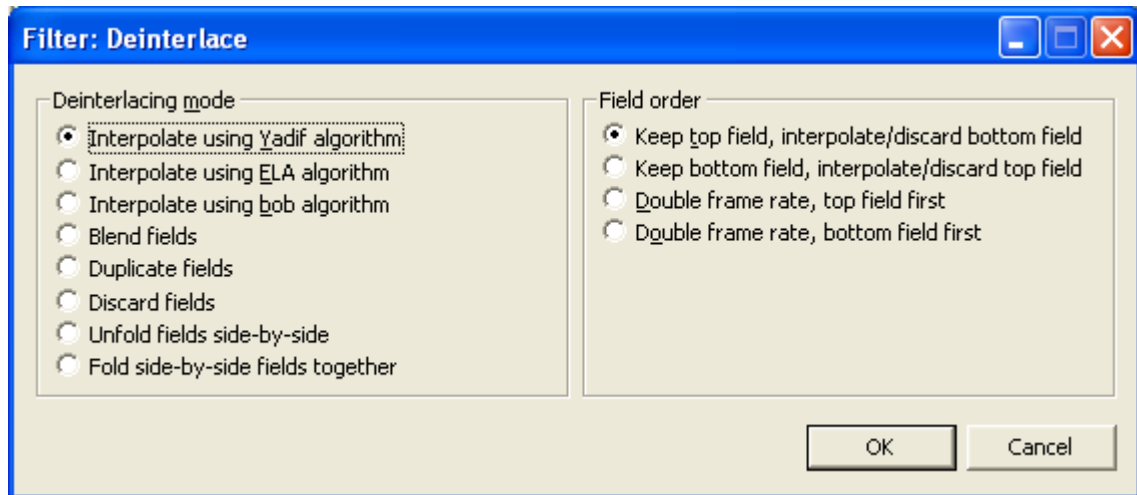
In order to de-interlace the video, please select:

Video/ filter/ add /deinterlace and always choose **“Interpolate using Yadif algorithm”**. If your video is PAL choose **“Keep top field, interpolate/discard bottom field”** and if your video is NTSC choose **“Keep bottom field, interpolate/discard top field”**.

Figure 7 Before de-interlace(on the left) and after de-interlace(on the right)



Figure 8 The various de-interlace techniques in VirtualDub, Yadif is selected



2.4 Levels adjustments

On most video footage black and white tend to be washed out: black appears dark grey and white looks like light grey.

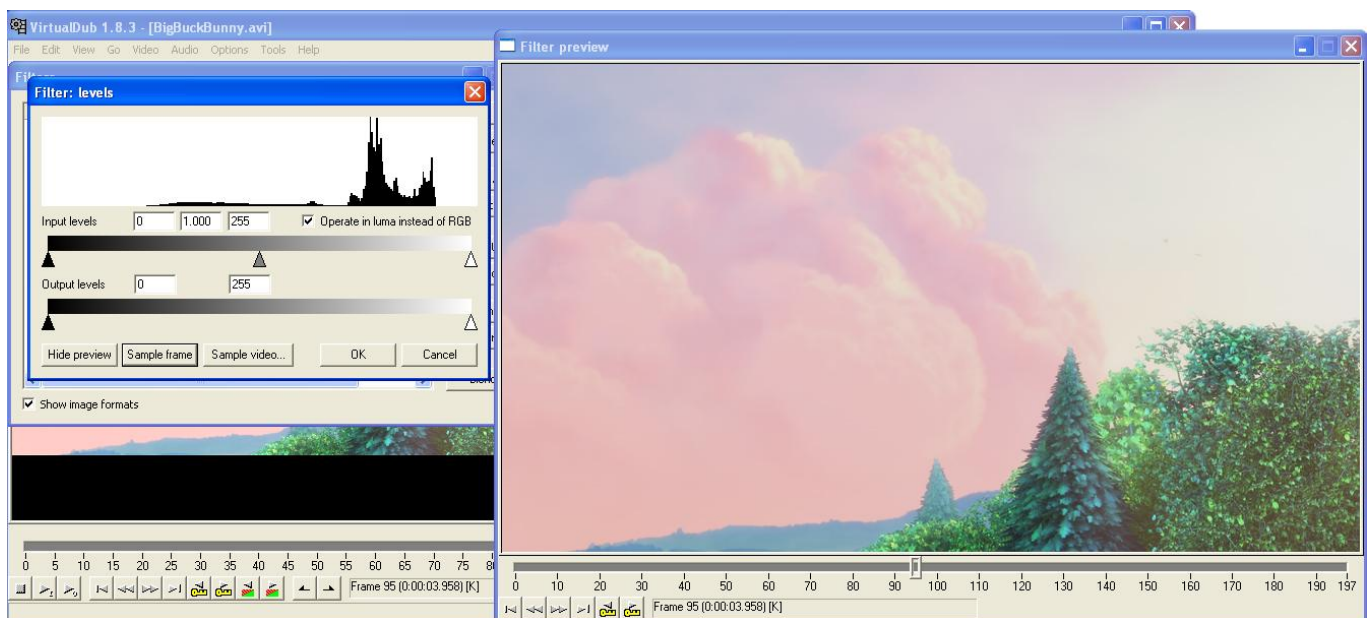
In order to fix this issue we need to manually modify the video levels.

First, load the level filter:

Video/ filter/ add/ levels

We'll start with adjusting the white level: Select "Preview" then choose a frame with a lot of white and select "Sample frame" in order to display the histogram.

Figure 9 Levels adjustments in VirtualDub, Step 1



If white colors were really 100% white, the right side of the histogram (representing the white level) would touch the right side of the dialog box. As we can see it doesn't: we need to manually move the white cursor so that it touches the first value of white on the right side of the histogram.

Figure 10 Levels adjustments in VirtualDub, Step 2

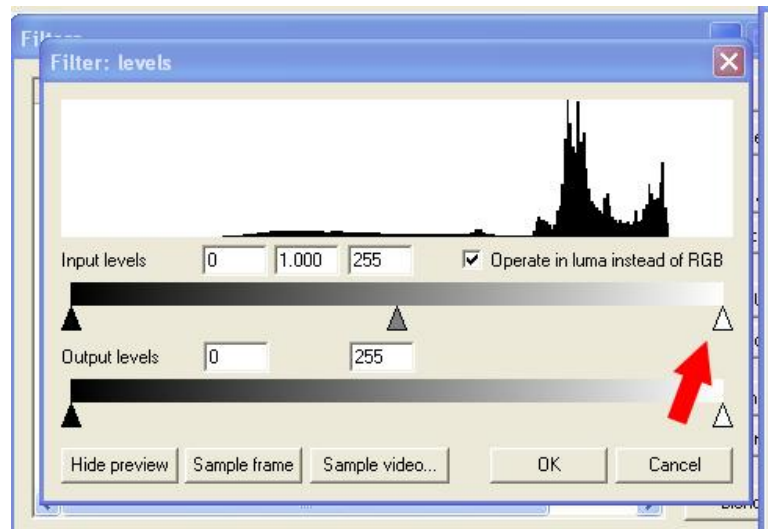
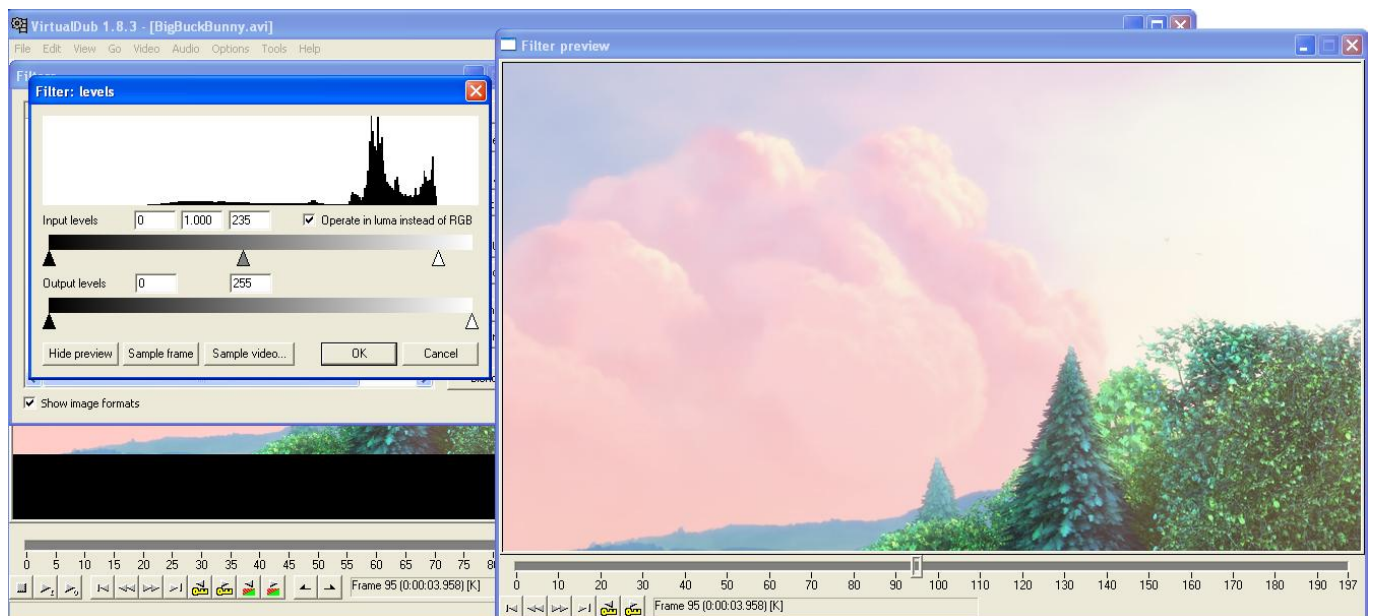
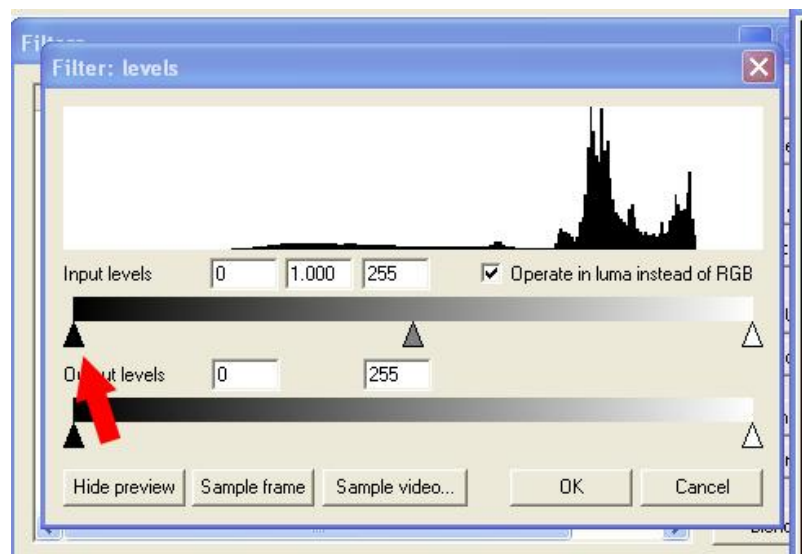


Figure 11 Levels adjustments in VirtualDub, Step 3



Now we just have to do the same with the black levels on the left side of the histogram (in this case it is better to choose a dark image in the video).

Figure 12 Levels adjustments in VirtualDub, Step 4



2.5 Resize

You often have to resize the original video to the target resolution.

Note: NOTE: If the video source is already in the target resolution, you can skip this step.

To resize your video, go to:

Video/ filter/ add/ resize

In **FILTER MODE** you should always select **Lanczos 3**

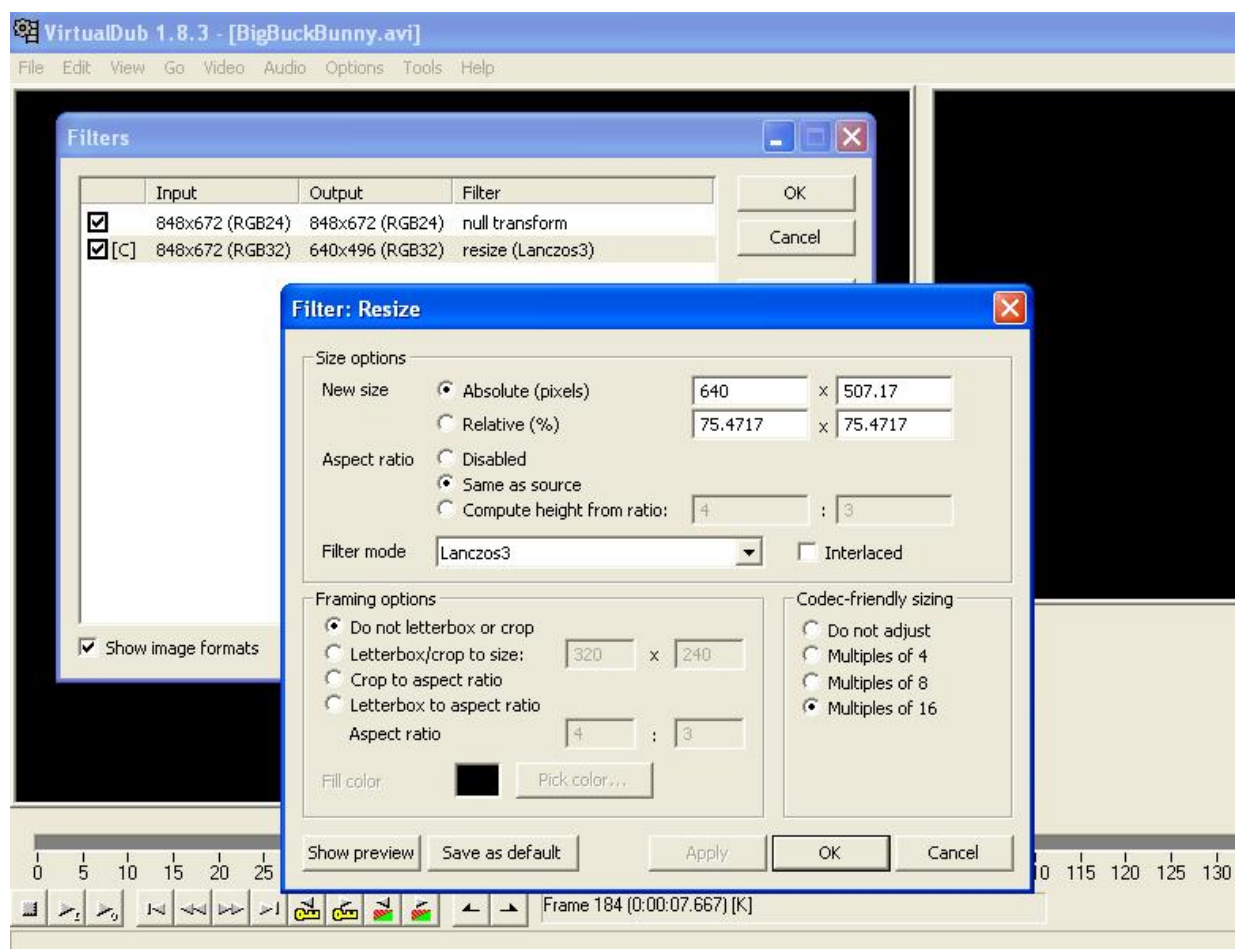
In **ASPECT RATIO** you should select **Same as source**

The example described below show how to perform a resize for a target display that has VGA (640x480) resolution.

The goal is to obtain the native target resolution to do full screen playback:

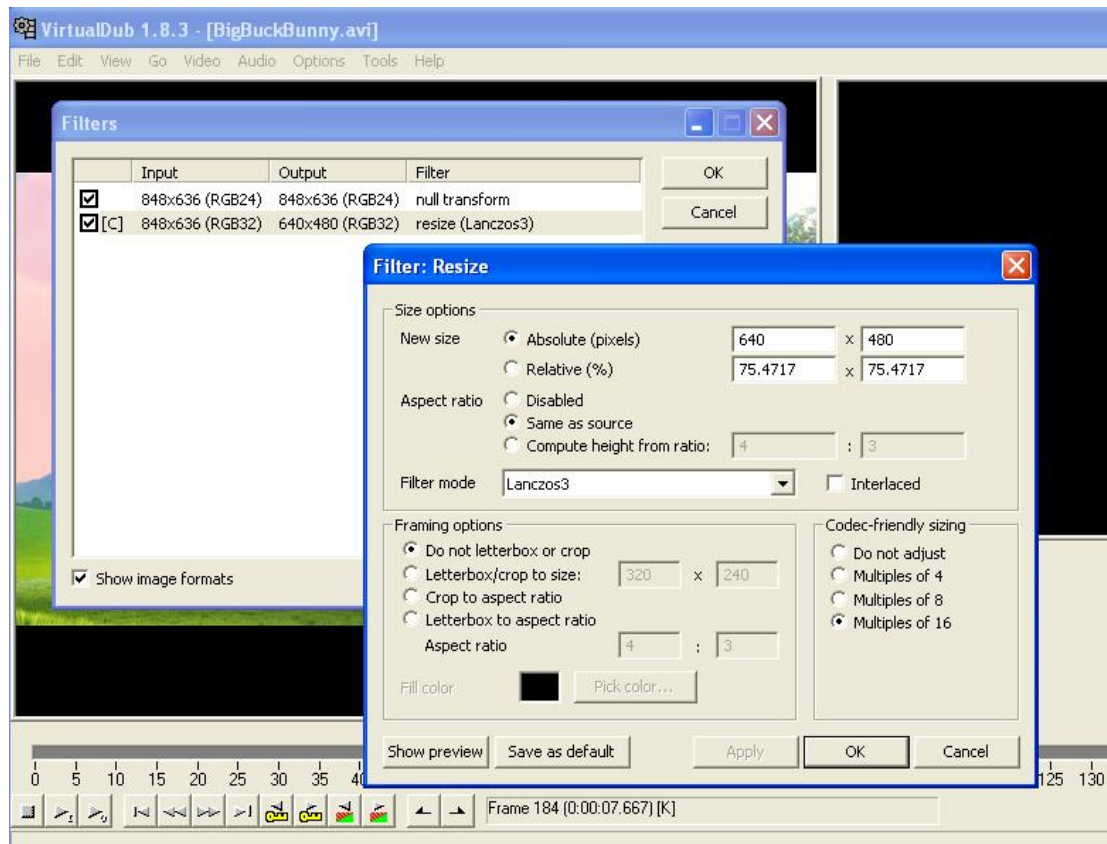
- Select the width of your video(640 pixels in our example): the corresponding height that respects the movie aspect ratio will automatically been computed.
- You must select "Multiples of 16".
- Press OK
- It will now select a resolution (for example, 640x496.) Note that it is OK if the video height is not exactly 480 pixels, as you can crop this again later.

Figure 13 Resizing in VirtualDub, Step 1



To have the desired resolution (640x480), you probably need to crop the video to obtain the desired height. See the cropping section (earlier in this guide) to do so.

Figure 14 Resizing in VirtualDub, Step 2



3 Processing 3D resources

Source videos need to be prepared in specific ways in order to do stereoscopic playback on the CTR platform.

There are 3 ways to prepare stereoscopic video content for the CTR platform:

- Using a **top to bottom** layout
- Using a **side by side** layout
- Using an **interleaved** layout

Figure 15 Exemple of a top to bottom layout



Figure 16 Example of a side by side layout

Note: While it is possible to prepare and encode a Mobiclip video using either **top to bottom** or **side by side** layout, it is not the recommended way to obtain the best possible quality stereoscopic video out of Mobiclip VFW codec. These 2 layouts are nevertheless supported because they can be easier to use for example if the video to encode is already available with this layout. What is recommended is to prepare the video using an **interleaved** layout. This is what will be described below.

The most efficient way to prepare a stereoscopic video is to mix the video data of left and right eyes into one video stream in an interleaved fashion. This what is called an **interleaved** layout.

Example:

left eye frame 1, right eye frame 1, left eye frame 2, right eye frame 2, ...

The resulting interleaved video will have the same resolution as the left and right eye source videos and its frame rate will be the double of them.

This stereoscopic layout has several benefits:

- each “eye frame” is encoded with reference to both eyes previous and current frames, which improves encoding efficiency (and thus reduces bit-rate and CPU load) as the left and right eye images are obviously correlated. You can expect around 20% quality/compression ratio improvement compared to both the **top to bottom** and **side by side** layouts.
- by decoding a single interleaved stream you can reduce the memory load and improve cache memory efficiency.

Before encoding a 3D video using the Mobiclip VFW codec for NINTENDO CTR you need to pre-process the video to have the format described above.

To perform that task we'll use later on in this document the Avisynth tool.

To use Avisynth you only have to write Avisynth scripts, as text files with the .avs extension.

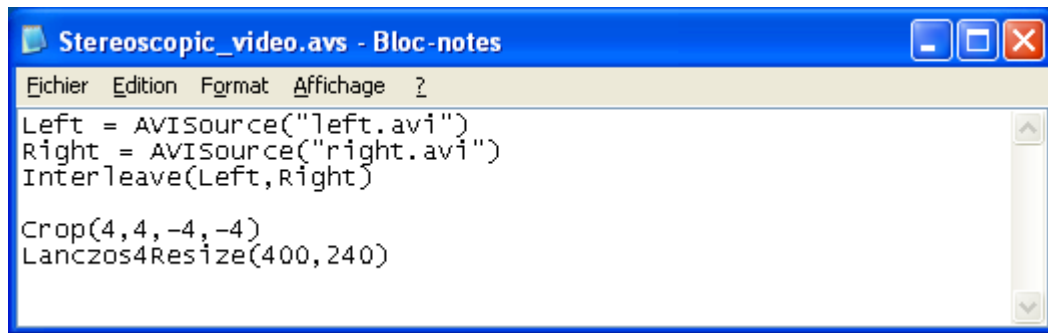
You can later open these .avs files with any video editing tool like VirtualDub, they will behave like an AVI file with uncompressed RGB video data.

Note: It is strongly recommended to have an intermediate video file (or an Avisynth script file) perfectly processed before starting encoding: it is easier if you have to encode the same video several times.

3.1 Case 1: One source video for each eye

In this case you have two physical different video files, one for each eye, with the same resolution and frame rate.

Figure 17 Avisynth script that interleaves two different source videos



In this AviSynth script we have two inputs, one for the left eye and one for the right eye and we want to interleave **Left** and **Right**. You can add **Crop** and **Resize** after the **Interleave** command.

Note: The two video inputs must have exactly the same width and height: the .avs script will crash if the two inputs have different resolutions.

3.2 Case 2: One video containing both left and right eyes

You may encounter video sources with both left and right eyes inside in the same video stream frame. Each eye is stretched to fit half resolution of the source. For that reason you must be very careful with the source display aspect ratio.

Figure 18 Example of 3D source video with left and right eyes placed side by side



In this example the resolution of the source is 1920x1080 and is only 960x1080 for each eye. In this case the Avisynth processing is a bit different: before interleaving eyes you need to separate them by cropping the video:

Figure 19 Avisynth script that interleaves left and right eye image from the same source video

```
Strereoscopic_video.avs - Bloc-notes
Fichier Edition Format Affichage ?
Left = AVISource("video.avi").Crop(0,0,-960,-0)
Right = AVISource("video.avi").Crop(960,0,-0,-0)
Interleave(Left,Right)

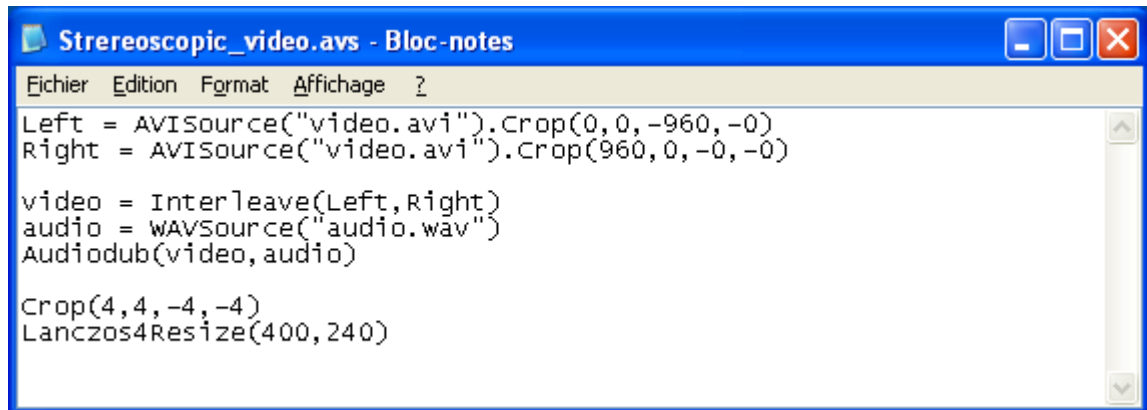
Crop(4,4,-4,-4)
Lanczos4Resize(400,240)
```

Note: the syntax of the crop filter in Avisynth is: **Crop(left,top,-right,-bottom)**

The first crop will crop 960 pixels from the right side, and the second one will crop 960 pixels from the left side. In this case you have to be very careful with the aspect ratio of the source and the final aspect ratio.

If you need to add an audio track to your script you can use the syntax below:

Figure 20 Add audio track inside the Avisynth script



3.3 About pre-rotating video

A way to lower FCRAM memory usage a bit is to pre-rotate all images of a stereoscopic interleaved or 2D video sequence by 90 degree clockwise.

You would obtain an image like the figure below:

Figure 21 Example of pre-rotated video



Additional information on using pre-rotated movies can be found in the *CTR-Mobiclip_SDK_Programming_Manual-en_US.pdf* document.

3.4 About CTR liquid crystal display frequency and video frame rate

Most commonly used video movies frame rates are 30 fps, 29.97 fps, 24 fps or 25 fps.

On the other hand the CTR liquid crystal display (LCD) frequency is 59.831 Hz (that means the display frame rate is 59.831 fps).

If LCD frequency is not a multiple of the target movie frame rate, some movie video frames will be irregularly displayed several times and others not, in order to match the LCD refresh rate.

As a consequence the user will experience choppiness when displaying video frames during playback. This choppiness will be more visible for videos where their frame rate multiples are significantly different from the LCD refresh rate: for example choppiness will be less visible on 30 fps and 29.97fps movies than on 24fps and 25fps movies.

While this is not a critical issue and can be left as is for most scenarios, this chapter will describe various solutions to fix it.

3.4.1 Changing movie frame rate value

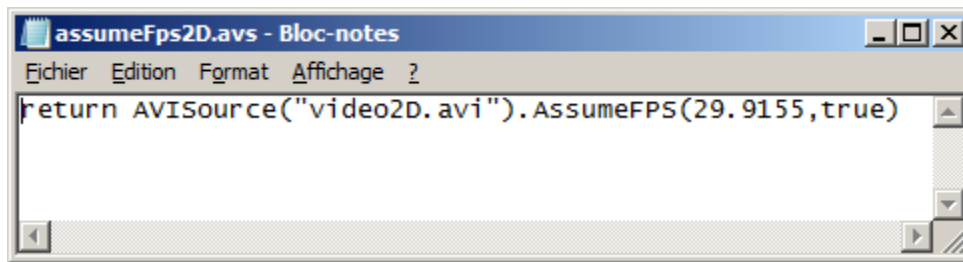
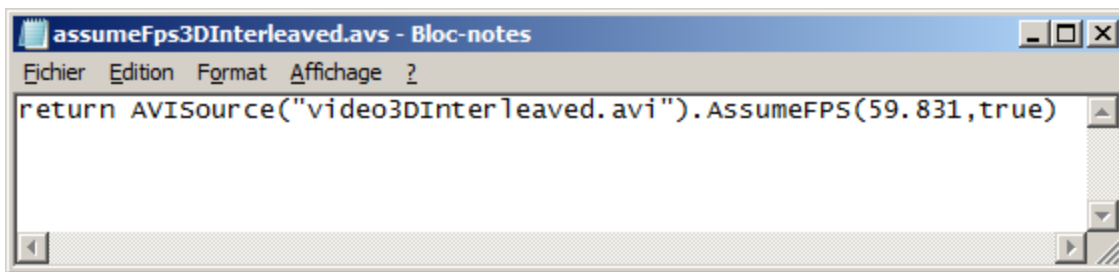
This solution consists of changing the video movie frame rate value to a sub-multiple of the LCD refresh rate, that is 59.831 fps or 29.9155 fps for example.

Associated audio tracks should have their frequency value modified or should be re-sampled to compensate the frame rate value change.

This solution is not recommended for video movie frame rate values that are not close to a sub-multiple of the LCD refresh rate, that is 24 fps, 25 fps or 50 fps for example.

On the contrary this solution works well with 29.97 fps, 30 fps, 59.94 fps or 60 fps for example.

A way to do that is to use an Avisynth script like in the following examples.

Figure 22 Frame rate change to 29.9155 fps**Figure 23 Frame rate change to 59.831 fps of a 3D interleaved movie**

In these examples the Avisynth filter *AssumeFPS* is used.

It only changes the video frame rate and the audio frequency values of the movie.

As no new video frames are inserted in the movie you can apply this filter on an already interleaved 3D video.

Please have a look at the *Avisynth Filters Information Reference* chapter.

3.4.2 Inserting video images to match the LCD refresh rate

This solution consist of inserting new video images at some points in the video. These new video images content are a blend of the preceding and following video images.

Blended images are rather difficult to compress by video codecs. As a consequence the compressed video will be sensibly bigger and/or of lower quality than using the original video.

A way to do that is to use an Avisynth script like in the following examples.

Figure 24 Frame rate conversion to 29.9155 fps

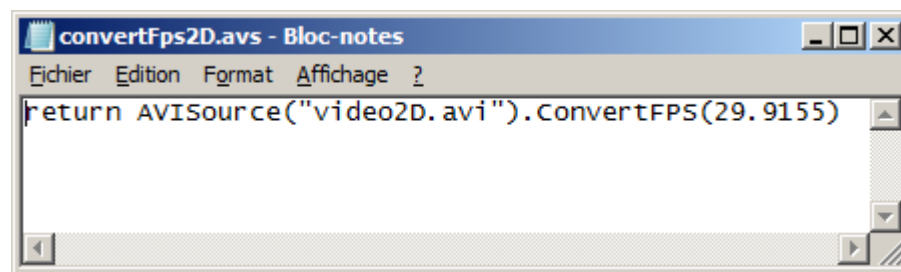
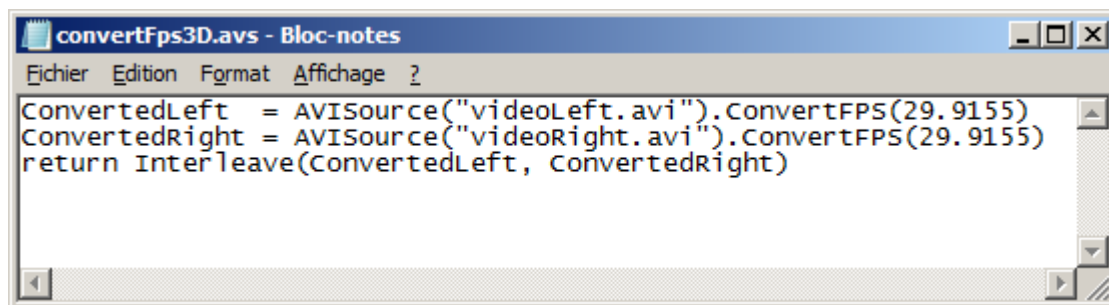


Figure 25 Frame rate conversion to 29.9155 and interleaving of left/right eye image



In these examples the Avisynth filter *ConvertFPS* is used.

It adds blended video images at some points in the video to target the wanted frame rate.

As new video images are inserted this method cannot be used on an already 3D interleaved video.

Please have a look at the *Avisynth Filters Information Reference* chapter.

4 Avisynth Filters Information Reference

This chapter only gives useful information on the most important Avisynth filters that are used in this document, it is not meant to be a reference of all used Avisynth filters.

Please have a look at the official Avisynth website for the latest information on that topic:

http://avisynth.org/mediawiki/Main_Page

4.1 AssumeFPS filter

```
AssumeFPS(clip movie, float fps, bool sync_audio)
```

The *AssumeFPS* filter changes the frame rate without changing the frame count (causing the video to play faster or slower).

It only sets the *framerate* parameter. If *sync_audio* (which is false by default) is true, it also changes the audio sample rate to match the duration of the video, the pitch of the resulting audio gets shifted.

For more details on this filter please see its official Wiki documentation at the following link:

<http://avisynth.org/mediawiki/AssumeFPS>

4.2 ConvertFPS filter

```
ConvertFPS(clip movie, float fps)
```

The *ConvertFPS* filter attempts to convert the frame rate of *movie* to *fps*, providing a smooth conversion with results similar to those of standalone converter boxes. The output will have almost the same duration as clip, but the number of frames will change proportional to the ratio of target and source frame rates.

For more details on this filter please see its official Wiki documentation at the following link:

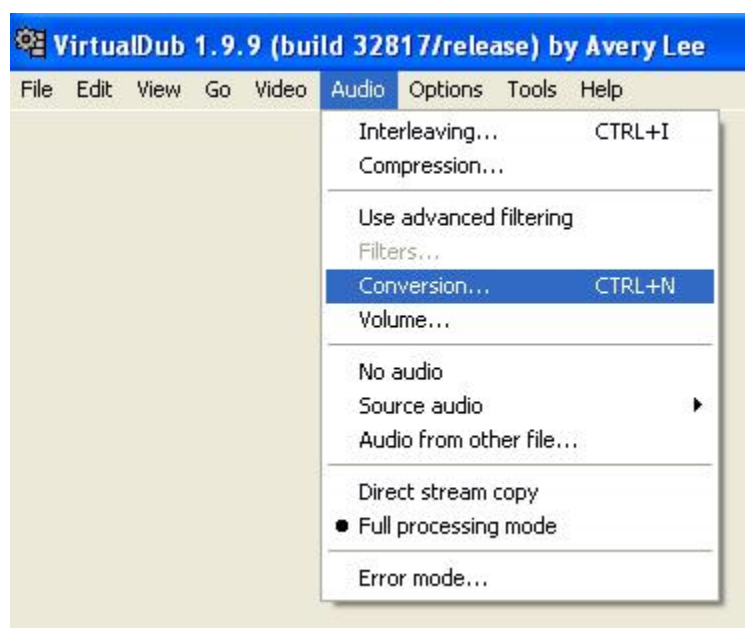
<http://avisynth.org/mediawiki/AssumeFPS#ConvertFPS>

5 Audio Encoding

You cannot modify the sampling rate and the mono/stereo settings directly in the *MoflexTool* program but you can do that in an audio/video tool like VirtualDub.

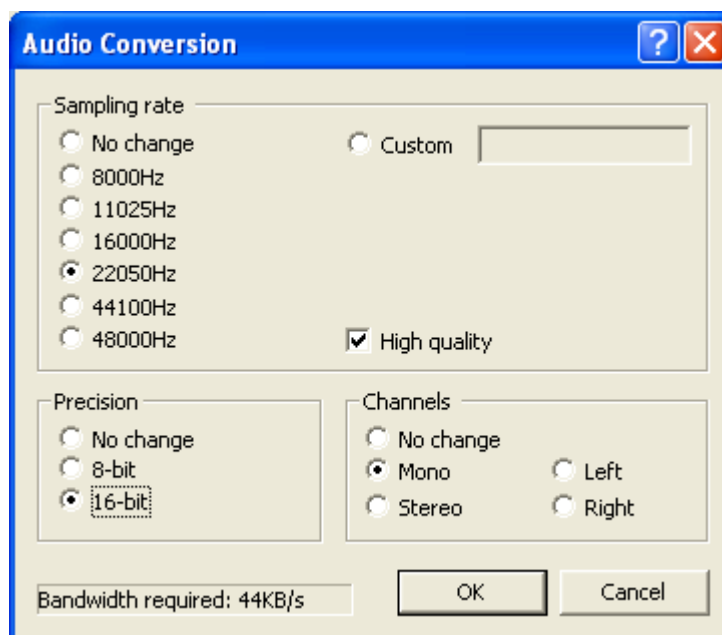
Load your movie into VirtualDub. If the audio is not included in the media file, select “Audio” menu, “Audio from other file...” and select the .wav audio track on your computer. Then, click on the “Audio” menu again, make sure the “Full processing mode” is selected and click on “Conversion”:

Figure 26 Audio conversion in VirtualDub



Note: It is recommended to normalize the audio track before doing any destructive processing (down-sampling, down-mixing).

Figure 27 Audio conversion settings



If you want to convert your audio track to another sampling rate, you can use custom settings and write the frequency you want. Make sure “High quality” and “16-bit” are checked.

6 Mobiclip Video for Windows(VFW) codec description

This chapter explains how to access and use the Mobiclip VFW codec configuration menu.

It then explains which kind of encoding scheme to use depending on different movie playback scenarios.

6.1 Access the VFW codec configuration menu

You have two ways to access to the configuration menu of the Mobiclip Video for Windows codec:

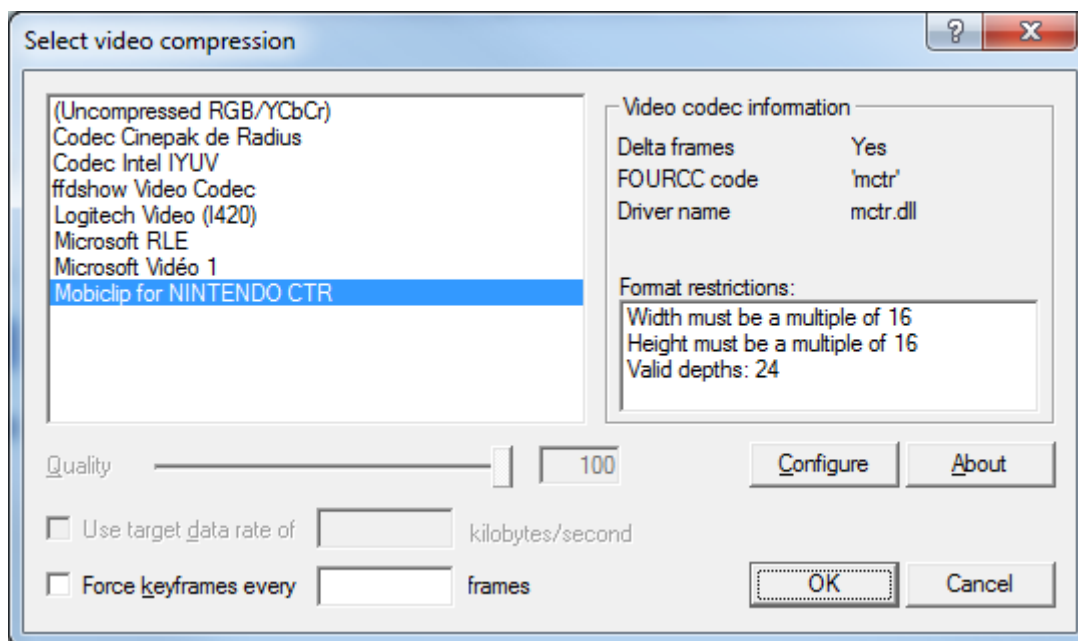
- From the Windows **Start menu**, go to **All Programs** and in **Mobiclip Tools for NINTENDO CTR** click on **Mobiclip VFW codec for CTR Configuration**.
- From a video editing tool.

Below is described a way to configure the Mobiclip Video for Windows codec from VirtualDub.

6.1.1 Selecting the Mobiclip VFW codec for NINTENDO CTR

In VirtualDub, open the **Video** menu, click **Compression** and choose Mobiclip for NINTENDO CTR.

Figure 28 VirtualDub VFW codec selection dialog box



6.1.2 Opening the Mobiclip VFW codec for NINTENDO CTR configuration dialog

Click on the **Configure** button

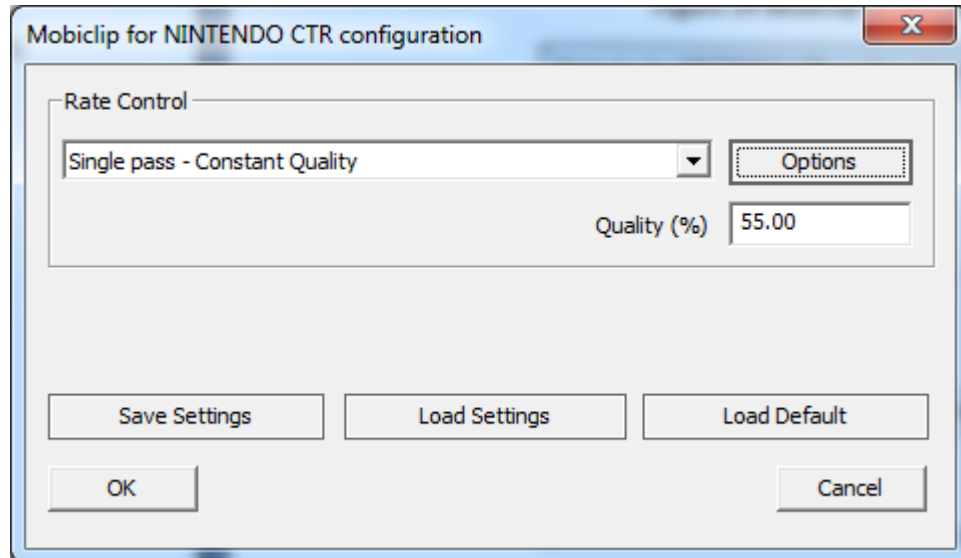
6.1.3 Tuning the parameters of the Mobiclip VFW codec for NINTENDO CTR

Set the parameter values.

You will find below a complete description of the configuration dialog.

6.2 VFW codec configuration reference

Figure 29 Mobiclip VFW codec configuration dialog box



6.2.1 Encoding Type

In the **Rate Control** section you can choose what type of encoding algorithm you want to use.

There are five different choices.

6.2.1.1 Single Pass – Constant Quality

This is the default method. The video is encoded based upon the quality parameter. Valid values are in the range 0 to 100, and the default value is 55.

Quality value of 0 is the lowest standard of picture quality that can be practically used.

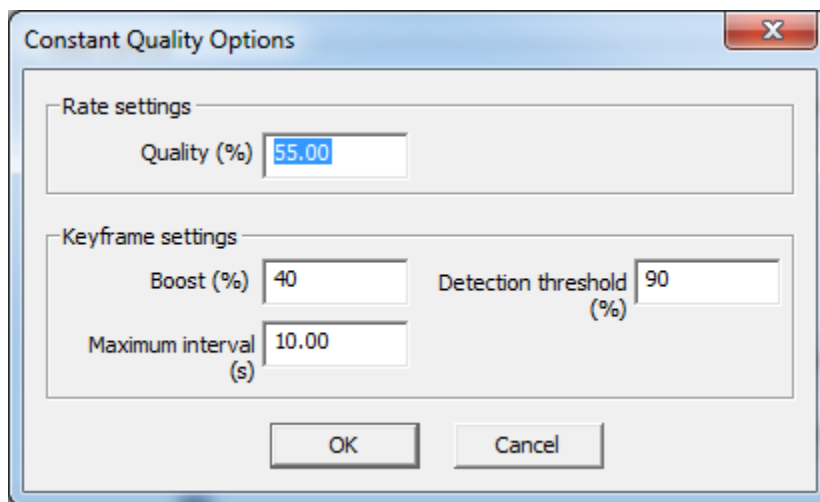
When **Quality** has a value of 100 you will not see improvement any more even though you lower the compression rate more.

A **Quality** value of 55 lets you achieve good video quality with a reasonable video size.

Note: **Quality** value range and perceived quality are based on developer's standard and are not quantitative numerical values.

This method is only available in a **Single Pass** encoding type. When *Options* button is pressed the following dialog box appears:

Figure 30 Mobiclip VFW codec Constant Quality Options dialog box



1. Quality

The quality parameter described above.

2. **Boost**

This parameter is used to boost the quality of the encoded keyframes of the video. This often results in a quality increase in the video sequences after keyframes, and its default value is 40.

Note: !!! WARNING: don't modify this parameter except if you are an expert user !!!

3. **Detection threshold**

This is a tuning parameter for scene change detection.

The default value is 90, which means that if more than 90% of the image has changed it will be flagged as a keyframe.

If you have static parts in your video frames like permanent black band, you should lower this value. For example if there is only 70% of your images which is changing (30% are permanently black) you should put $70\% * 90\% = 63\%$ as the parameter.

4. **Maximum Interval**

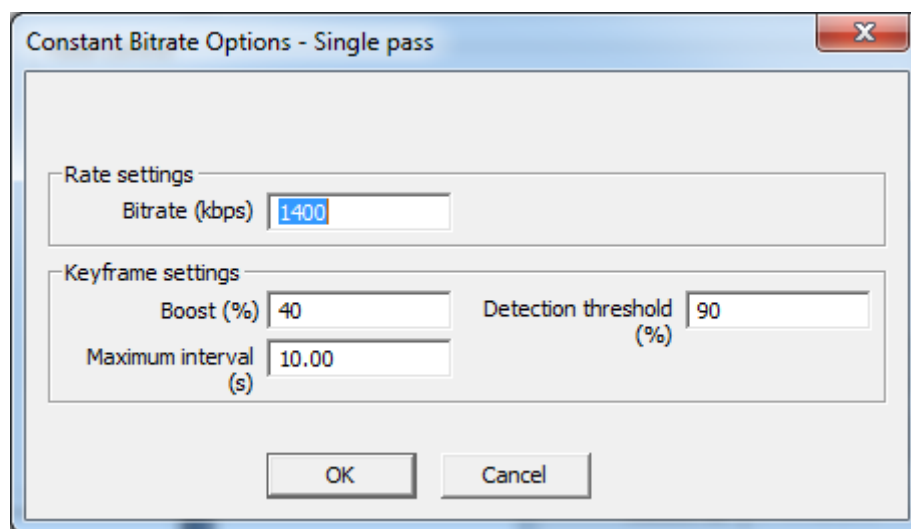
This is the maximum interval where the video encoder is forced to put keyframes in your video. Valid values (in seconds) are decimals in the range 0.0 to 3600.0 and the default value is 10.0 seconds, which means that the video encoder will be forced to put a keyframe in your video every 10 seconds interval, even if no scene change has been detected.

The maximum number of keyframes that can be managed by the Moflex file format is $2^{32}-1$.

6.2.1.2 Single Pass – CBR

The video encoder uses single pass constant bitrate encoding type. The **Bitrate** parameter is given in kilobits per second, valid values are in the range 1 to 20000 and the default value is 1400. When **Options** button is pressed the following dialog box appears:

Figure 31 Mobiclip VFW codec Constant Bitrate Options dialog box



1. Bitrate

The bitrate parameter described above.

2. Boost, Detection threshold, Maximum Interval

They have the same meaning as in the *Single Pass – Constant Quality* encoding scheme. The default value for *MaximumInterval* is 10 seconds.

Note: Please note that CBR does not mean that the bitrate will not change at all in the encoded video data, in fact there can be moderate variations of bitrate. This is because most of the time video CBR controllers are specified so that a simulated buffer that is filled at constant bitrate should not get empty nor full. In audio codec terminology, this type of rate control is called an ABR (standing for average bitrate).

6.2.1.3 Multipass - VBR - First Pass

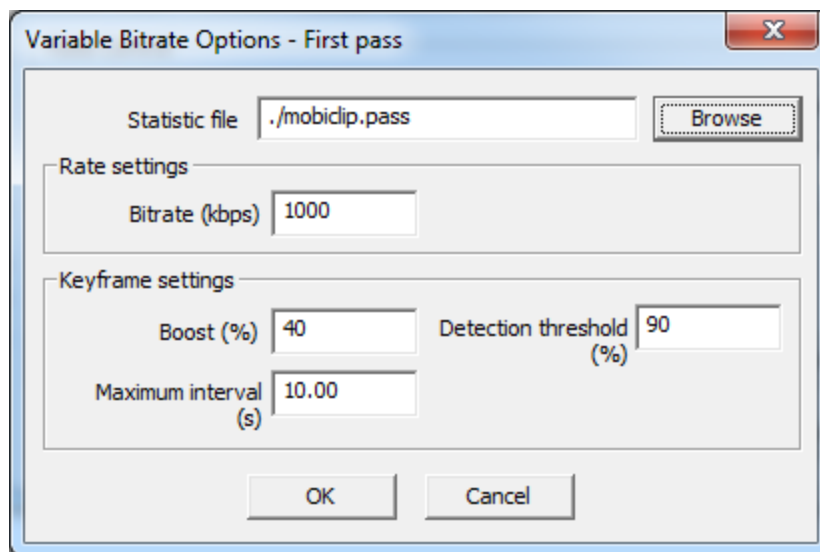
The first pass of a variable bitrate multipass video encoding type is used to gather statistics about the video to optimize the next passes (Multipass - VBR - Nth Pass). The multipass VBR encoding scheme is useful when you want the best video quality for a given target size.

Given the duration of the movie and the desired target size, you can calculate the corresponding average bitrate value that is required, like below:

Average bitrate (kbps) / 8 * duration (seconds) = video target size (KBytes)

The **Bitrate** parameter is given in kilobits per second, valid values are in the range 1 to 20000 and the default value is 1000. When **Options** button is pressed the following dialog box appears:

Figure 32 Mobiclip VFW codec Variable Bitrate Options – First pass dialog box



1. Stat File

This lets you configure the location and the name of the file which will hold the statistics used for multi-pass encoding. The default file is './mobiclip.pass'.

2. Bitrate

The average bitrate parameter described above.

3. Boost, Detection threshold, Maximum Interval

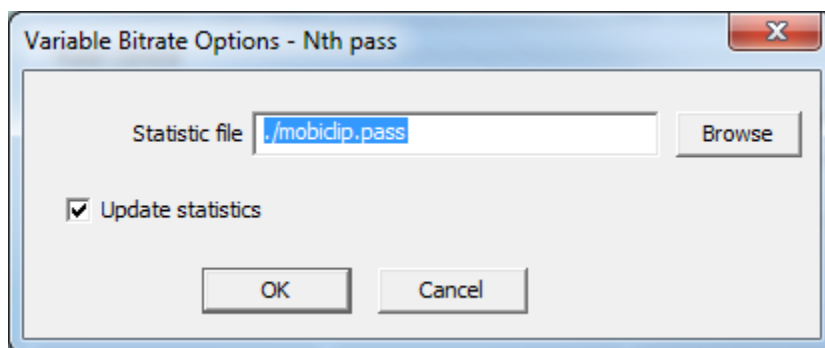
They have the same meaning as in the *Single Pass – CBR* encoding scheme.

6.2.1.4 Multipass - VBR - Nth Pass

The successive passes of a variable bitrate multipass video encoding type are used to optimize the video quality, once statistics have been gathered by the first pass.

This process can be executed any number of times to better target the desired average bitrate or file size. It should be executed at least once to have a usable video target, and if you call it twice you will be very close to the targeted bitrate. When *Options* button is pressed the following dialog box appears:

Figure 33 Mobiclip VFW codec Variable Bitrate Options – Nth pass dialog box



1. Stat File

This is the same parameter described previously.

2. Update Statistics

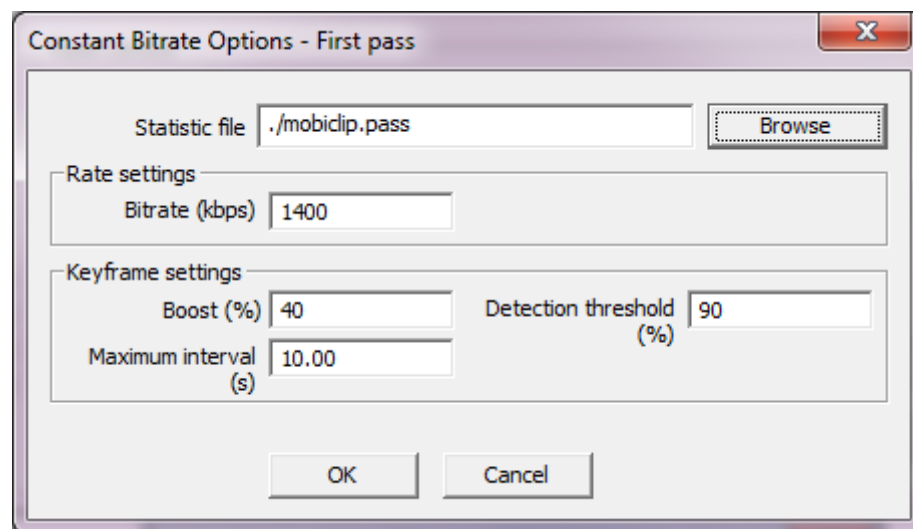
Check this box if you want the encoding statistics to be updated during this pass, and it is checked by default. This option is useful if you want to encode a movie using the Multipass – VBR - Nth Pass encoding type more than one time.

6.2.1.5 Multipass - CBR - First Pass

The first pass of a constant bitrate multipass video encoding type is used to gather statistics about the video to optimize the next passes (Multipass - CBR - Nth Pass.) This encoding scheme will give you better video quality than the CBR single pass scheme.

The *Bitrate* parameter is given in kilobits per second, valid values are in the range 1 to 20000 and the default value is 1400. When *Options* button is pressed the following dialog box appears:

Figure 34 Mobiclip VFW codec Constante Bitrate Options – First pass dialog box



1. **Stat File**
2. **Bitrate**
3. **Boost**
4. **Detection threshold**
5. **Maximum Interval**

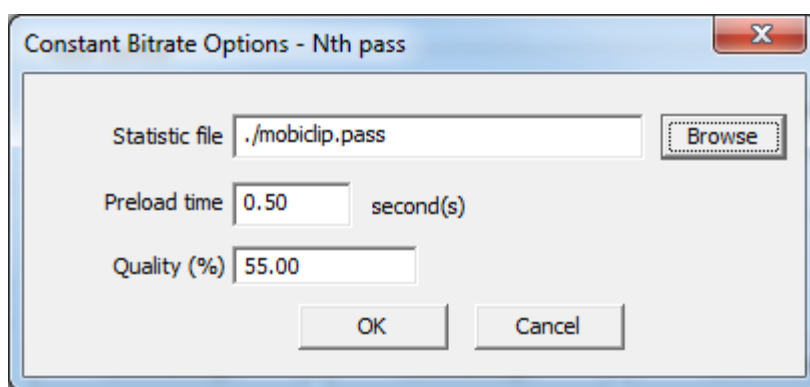
They have the same meaning as described above.

6.2.1.6 Multipass - CBR - Nth Pass

The successive passes of a constant bitrate multipass video encoding type are used to optimize the video quality, once statistics have been gathered by the first pass.

This can be executed any number of times to better optimize the video quality given the desired bitrate. Calling it once gives already very good results. When *Options* button is pressed the following dialog box appears:

Figure 35 Mobiclip VFW codec Constante Bitrate Options – Nth pass dialog box



1. Stat File

This parameter is already described in previous sections.

2. Preload Time

This value is the time that the decoder should take to preload the stream data before beginning the playback. The higher the value, the better the encoder will be able to optimize the quality of the video. Valid values are between 0 and 3600 seconds, the default value is 0.5 second.

3. Quality

This value is the targeted maximum quality. This field has the same meaning as in the "Single Pass – Constant Quality" encoding scheme. Valid values are in the range 0 to 100, default being 55.

The encoder will not waste space to encode video frames whenever the targeted quality can be achieved with a lower data rate. As a consequence the resulting average bitrate is usually significantly lower than the one initially specified.

6.2.2 Multipass encoding howto

Here are the basic steps to follow for multipass encoding.

1. First encoding

- Select "Multipass - First Pass(CBR or VBR)" from the Mobiclip VFW codec configuration dialog box. At this time you can change the place where to store the statistics file by pressing the **Option** button.
- Close the dialog box and perform the first encoding.

2. The successive encoding

- Select "Multipass - Next Pass(with same bitrate as the first encoding)" from the Mobiclip VFW codec configuration dialog box. Set the original source to the second encoding, not the AVI file generated from the first encoding. Be careful not to use the AVI file that was generated from the first encoding. You can change the place where to read/update the statistics file by using the **Option** button. Choose the **Stat File** created in the previous encoding pass. If you plan to perform more than two encoding passes, please check the **Update Statistics** check box.
- Close the dialog box and perform the successive encoding passes.

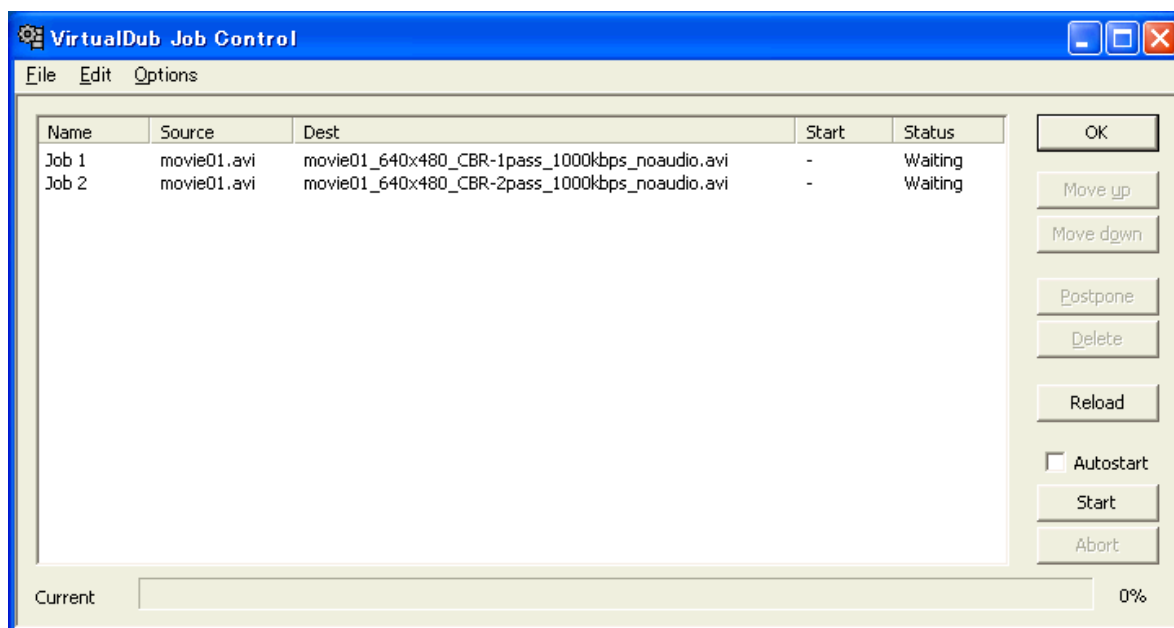
3. Utilizing job control

Please note that under VirtualDub tool you can do all these passes automatically by using its batch job feature.

To create and use batch jobs you have to do the following steps:

- For each encoding pass select the **File** menu, click on **Save As AVI** and check **Don't run this job now; add it to job control so I can run it in batch mode** before saving.
- If you are using VirtualDub V1.6.9, for each encoding pass select the **File** menu, click on **Queue batch operation** and **Save As AVI**.
- Click on **File** and then **Job Control** to manage and launch encoding batches.

Figure 36 VirtualDub Job Control Window



6.3 VFW codec configuration walkthrough

Configuring it can be a complex task, given all the encoding schemes and options provided by the Mobiclip VFW codec.

Please note however that the default parameter values are already tuned to obtain a good quality/compression tradeoff for all available encoding schemes.

This section explains how to choose the right encoding scheme depending on specific needs.

6.3.1 Constant Quality encoding scheme

It is a one pass encoding targeting a constant quality for all video frames.

This is the easiest way to encode a Mobiclip video and it is selected by default.

A value of 55% for the **Quality** parameter already gives a good compression/quality tradeoff: video compression artifacts are kept minimal and average bitrate sits between 500 and 800 Kbps for most of full screen stereoscopic videos (400x240 frames interleaved at 25-30fps).

The drawback of this method is that we don't have any control of the bitrate of the resulting movie. Some encoded video frames will be big (that is not compressed much) in order to target the requested quality and will be complex to decode (that is will take a lot of CPU resources to decode). These are the case of video frames that can't be predicted from previously encoded ones because they don't share much visual information; they are often called IFrames or Keyframes, PFrames being frames predicted from previously encoded ones.

When this happens on consecutive video frames (like action movie explosion sequences for example) the video bitrate will be locally very high and video decoding might take too much CPU resources on the CTR platform, resulting in audio playback stuttering due to audio buffer underflow.

This can also lead to higher FCRAM (main) memory usage when demuxing video frames (that is extracting the video frames from the Moflex content).

In the case of lossy video compression technologies (like is done in MPEG1/2/4, H264 ,Mobiclip and others), the bigger the video frames (we can also say the bigger the video bitrate) the more complex the video frames are to decode.

This is because all these video codecs use some kind of DCT algorithm to encode the video details and these algorithms are all very CPU expensive compared to a simple block prediction. More video frames are big, more they are encoded using these 'DCT like' algorithms and as such more they are complex to decode.

The higher the **Quality** value, the more likely you will encounter this problem.

The solution would not be to lower the **Quality** parameter value as that would decrease the quality of all video frames. Instead, switch to the CBR multi-pass encoding scheme that is explained below.

6.3.2 Multipass CBR encoding scheme

It is a multiple pass encoding scheme.

On the first pass, called "Multipass CBR - First Pass", a **Bitrate** value is given.

On the second and next passes, called "Multipass CBR - Nth Pass", a **Quality** value is given that have the same meaning as the constant quality encoding scheme. This pass can be called multiple times but most of the time calling it once is enough to obtain good average bitrate approximation.

The encoder will not waste space to encode video frames whenever the targeted **Quality** can be achieved with a lower data rate. As a consequence the resulting average bitrate is usually significantly lower than the one initially specified.

Also the encoder will avoid generating long sequences of heavy video frames as it has to fulfill the **Bitrate** value constraint. As a consequence the CPU resources needed to decompress the video on the CTR platform is upper-bounded, solving the issue that would be found using the constant quality encoding scheme.

With this encoding scheme you'll often obtain quality comparable to the constant quality scheme while ensuring that it will play fine on CTR platform, at the expense of an additional pass.

6.3.3 Multipass VBR encoding scheme

It is a multiple pass encoding scheme.

It targets an average bitrate with no constraints like in Multipass CBR encoding scheme in order to maximize video quality through all movie.

This encoding scheme might prove useful if you don't have much space left on ROM for your video resources as it allows you to target a specific size in bytes.

The drawback of this method is that we don't have any control of the bitrate of the resulting movie. Some encoded video frames will be big (that is not compressed much) in order to target the requested quality and will be complex to decode (that is will take a lot of CPU resources to decode). These are the case of video frames that can't be predicted from previously encoded ones because they don't share much visual information; they are often called IFrames or Keyframes, PFrames being frames predicted from previously encoded ones.

When this happens on consecutive video frames (like action movie explosion sequences for example) the video bitrate will be locally very high and video decoding might take too much CPU resources on the CTR platform, resulting in audio playback stuttering due to audio buffer underflow.

This can also lead to higher FCRAM (main) memory usage when demuxing video frames (that is extracting the video frames from the Moflex content).

In the case of lossy video compression technologies (like is done in MPEG1/2/4, H264 ,Mobiclip and others), the bigger the video frames (we can also say the bigger the video bitrate) the more complex the video frames are to decode.

This is because all these video codecs use some kind of DCT algorithm to encode the video details and these algorithms are all very CPU expensive compared to a simple block prediction.

More video frames are big, more they are encoded using these 'DCT like' algorithms and as such more they are complex to decode.

The higher the **Bitrate** value, the more likely you will encounter this problem.

The solution would not be to lower the **Bitrate** parameter value as that would decrease the quality of all video frames. Instead, switch to the CBR multi-pass encoding scheme that is explained below.

6.3.4 Single Pass CBR encoding scheme

It is a one pass encoding targeting a constant bitrate.

Bitrate will not change much through all encoded video frames.

This encoding scheme is useful when reading movies from tightly bandwidth constrained mediums (like when reading a Moflex content through an internet connection).

As it is not the case of ROM or MMC/SD/SDHC mediums this encoding scheme usage is not recommended as it would increase video size and/or decrease video quality.

7 Conversion to MoFlex file format

In order to convert an encoded Mobiclip Video from AVI to Moflex container format (with .moflex file extension), you need to use the MoflexTool program; the MoflexTool program replaces the old AviToMoflex program and has much more functionalities, such as generating MoFlex content with multiple audio and video tracks.

To ease the conversion process you can create text files with the .bat or .cmd extension that will contain all command line parameters. You'll then only have to double click on the .bat or .cmd file to start the process.

Note: Video tracks, audio tracks and timeline data (used to accurately seek in the Moflex file on video keyframes) are all managed as streams inside a Moflex file. The Moflex container format can manage up to 256 streams. Please note however that increasing the number of streams stored in a Moflex file will also increase its overall bitrate.

7.1 MoflexTool command line parameters description

Here is a description of the available MoflexTool parameters to generate a Moflex content:

- **-avi** AVIFILE: specifies the name of an input AVI file which must be in AVI file format and can contain multiple audio and video tracks. At least one AVI file must be specified and for each the following restrictions apply:
 - only Mobiclip video tracks are accepted, that is videos encoded with the Mobiclip VFW codec for NINTENDO CTR.
 - only mono or stereo 16 bits PCM audio tracks or no audio tracks are accepted. Valid audio frequencies are in the 8KHz to 48KHz range.
- **-wav** WAVFILE: specifies the name of an input WAV file which must be in WAV file format. Only mono or stereo 16 bits PCM formats are supported. Valid audio frequencies are in the 8KHz to 48KHz range.
- **-moflex** MOFLEXFILE: MOFLEXFILE must be in Moflex file format. Only used to retrieve information by using **-info** option. Do not use it with **-video** or **-audio** option.
- **-info** [VOPTION = VALUE]: prints to screen the information on all media files that are accepted by the tool:
 - if the **-out** parameter is not used at the same time, information about all files declared with either **-avi**, **-wav** or **-moflex** parameters is printed to screen and no Moflex file is generated.
 - if the **-out** parameter is used at the same time, information about newly generated Moflex file is printed to screen.
 - VOPTION can be:
 - **csv**: Dump frames information to a CSV file. Value can be **true** or **false** (default is false).
- **-video** [VOPTION = VALUE]: adds a Mobiclip video track to the output Moflex file. VOPTION can be:
 - **file**: file index, that is the index in the list of files declared with either **-avi**, **-wav** or **-moflex** parameters, starting from 0. VALUE is 0 by default.
 - **stream**: stream index, that is the index of the video stream inside the file selected by **file** option. VALUE is 0 by default.
 - **rotation**: describes clockwise rotation in degrees of video before its encoding. VALUE can be **none**, **90**, **180**, **270**. If **stereo** is specified and **rotation** is not, VALUE will be **none**. If neither **stereo** nor **rotation** is specified, VALUE will be judged in the order below:
 - if video resolution is 240x400 or 240x320, VALUE will be **90**.
 - if video resolution is 240x800, VALUE will be **90**.
 - if video resolution is 480x400, VALUE will be **90**.
 - for other video resolutions VALUE will be **none**.

- **stereo**: describes the stereoscopic layout, if any. VALUE can be **none**, **interleaved**, **toptobottom** or **sidebyside**. If **rotation** is specified and **stereo** is not, VALUE will be **none**. If neither **stereo** nor **rotation** is specified, VALUE will be judged in the order below:
 - if video frame rate is higher than 30, VALUE will be **interleaved**.
 - if video resolution is 400x480, VALUE will be **toptobottom**.
 - if video resolution is 240x800, VALUE will be **toptobottom**.
 - if video resolution is 800x240, VALUE will be **sidebyside**.
 - if video resolution is 480x400, VALUE will be **sidebyside**.
 - for other cases VALUE will be **none**.
- **firsteye**: describes the encoding eye order. VALUE can be **left** or **right**. VALUE is **left** by default. If stereo option is none this option will be ignored.
- **-audio** [AOPTION = VALUE]: adds an audio track to the output Moflex file. AOPTION can be:
 - **file**: file index, that is the index in the list of files declared with either **-avi**, **-wav** or **-moflex** parameters, starting from 0. VALUE is 0 by default.
 - **stream**: stream index, that is the index of the audio stream inside the file selected by **file** option. VALUE is 0 by default.
 - **format**: output audio format. VALUE can be **pcm**, **adpcm**, **dsp-adpcm**, **fa**. VALUE is **adpcm** by default.
- **-timeline** [TOPTION = VALUE]: adds timeline stream related to a given video or audio stream. A timeline stream stores information about the position of keyframes in the movie, this allows to precisely seek into a Moflex file. In case of an audio stream, the timeline stream stores the audio duration rather than the keyframes. TOPTION can be:
 - **file**: file index, that is the index in the list of files declared with either **-avi**, **-wav** or **-moflex** parameters. VALUE is 0 by default.
 - **type**: type of stream, that can be **video** or **audio**. VALUE is video by default.
 - **stream**: stream index, that is the index of the video stream inside the file selected by **file** option. VALUE is 0 by default.
- **-out**: specifies the name of the output Moflex file. If neither this parameter nor **-info** parameter are used, the name of the first input file (declared with **-wav**, or **-avi** parameter) is used with its extension replaced by .moflex string.
- **-packetsize** SIZE: specifies the size of the biggest packet in the Moflex file, in bytes. If nothing specified, SIZE will be 4096 by default. **WARNING**: this option should be restricted to advanced usages.
- **-usefixedpacketsize**: Moflex file will be generated with fixed packet size. Padding data can be added if necessary. **WARNING**: this option should be restricted to advanced usages.
- **-synchrofreq** USFREQ: Moflex syncho block frequency in microseconds. USFREQ is 15000000 microseconds (15 seconds) by default. **WARNING**: this option should be restricted to advanced usages.

Note: Several audio and video source files can be used as input at the same time by using several **-avi** or **-wav** parameters on the command line.

Note: If you need to specify multiple AOPTION, TOPTION or VOPTION options, you should specify them within a single **-audio**, **-timeline** or **-video** parameter, with each option separated by a space.

Note: The **-packetSize**, **-usefixedpacketSize** and **-synchrofreq** parameters descriptions have been added for reference only. Changing their default values should be restricted to advanced usages that are outside the scope of general Moflex content generation and playback.

Note: The audio frequency and the mono/stereo settings depend on the input audio file. If you want different settings you have to process the audio track using Virtual Dub before generating the Moflex file.

If no **-audio** parameters are specified then the Moflex file will be created without audio.

Note: At least a **-video** parameter must be defined if no **-info** parameter is used.

Note: Generated Moflex streams are numbered according to the order they appear on the command line (that is the order of the **-video**, **-audio** and **-timeline** parameters). These indexes range from 0 (included) to the total number of streams in the Moflex file (excluded).

Note: Only **-video** parameter **rotation** values **none** and **90** are used in practical playback situations.

We didn't find real usage benefits for the time being by using **180** and **270** values.

Note: When encoding audio in the **dsp-adpcm** format, `WaveCodecCtr.dll` is used to perform the conversion. The DLL must either be available in the regular search path or the `CTRSDK_ROOT` environment variable must be defined and point to the location of the CTR SDK. If the DLL cannot be found, an error message is displayed and the tool terminates.

7.2 MoflexTool command line examples

Below are several examples of the most common MoflexTool program usages.

7.2.1 Display MoflexTool help on the command line screen

```
MoflexTool
```

When executing the MoflexTool program from the command line and without any parameters, help is displayed on the screen.

7.2.2 Display AVI input file properties on the command line screen

```
MoflexTool -avi video.avi -info
```

This command is useful to retrieve AVI source file information before the conversion process to a Moflex file.

7.2.3 Display WAV input file properties on the command line screen

```
MoflexTool -wav track.wav -info
```

This command is useful to retrieve WAV source file information before the conversion process to a Moflex file.

7.2.4 Display existing Moflex input file properties on the command line screen

```
MoflexTool -moflex video.moflex -info
```

This command is useful to retrieve information about an already converted Moflex, either to check that the conversion went well, or to know its characteristics before using it in your CTR application.

7.2.5 Display a combination of AVI, WAV and Moflex input files properties on the command line screen

```
MoflexTool -avi avifile.avi -wav wavfile.wav -moflex video.moflex -info
```

This shows how to retrieve information about several input files of different types in one single MoflexTool program call.

7.2.6 Retrieval of the first video track of an AVI file

```
MoflexTool -avi video.avi -video
```

The resulting Moflex file has the following properties:

- the first video stream of the AVI file is stored in the target Moflex file.
- default settings for **-video** call is used (see **-video** parameter description for more details).
- output Moflex file name is video.moflex.
- no timeline is generated.

7.2.7 Retrieval of the first video track of an AVI file with timeline generation

```
MoflexTool -avi video.avi -video -timeline
```

The resulting Moflex file has the following properties:

- the first video stream of the AVI file is stored in the target Moflex file.
- default settings for **-video** call is used (see **-video** parameter description for more details).
- output Moflex file name is video.moflex.
- a timeline is generated for the video stream; this allows fast and accurate seeking to all keyframes of the associated video stream.

7.2.8 Conversion of the first audio track of an AVI file to a specific format

```
MoflexTool -avi video.avi -video -audio format=pcm
```

or

```
MoflexTool -avi video.avi -video -audio format=fa
```

The resulting Moflex files have the following properties:

- the first video stream of the AVI file is stored in the target Moflex file.
- default settings for **-video** call is used (see **-video** parameter description for more details).
- the first audio stream of the AVI file is stored in the target Moflex file as PCM (for the first example) or FastAudio (for the second example) format.
- output Moflex file name is video.moflex.
- no timeline is generated.

7.2.9 Use of an external audio track in WAV file

```
MoflexTool -avi video.avi -wav track0.wav -video -audio file=1
```

The resulting Moflex files have the following properties:

- the first video stream of the AVI file is stored in the target Moflex file.
- default settings for **-video** call is used (see **-video** parameter description for more details).
- the audio stream of the WAV file (that has an index order of 1, 0 being the AVI file index) is stored in the target Moflex file with default format.
- output Moflex file name is video.moflex.
- no timeline is generated.

7.2.10 Use of multiple audio tracks in WAV files

```
MoflexTool -avi video.avi -wav track0.wav -wav track1.wav -video -audio file=1  
-audio file=2
```

The resulting Moflex files have the following properties:

- the first video stream of the AVI file is stored in the target Moflex file.
- default settings for **-video** call is used (see **-video** parameter description for more details).
- the audio streams of the two WAV files (that have index order of 1 and 2 respectively, 0 being the AVI file index) is stored in the target Moflex file with default format.
- output Moflex file name is video.moflex.
- no timeline is generated.

7.2.11 Use of multiple audio tracks in WAV files and AVI files

```
MoflexTool -avi video.avi -wav track0.wav -wav track1.wav -video -audio file=0  
-audio file=1 -audio file=2
```

The resulting Moflex files have the following properties:

- the first video stream of the AVI file is stored in the target Moflex file.
- default settings for **-video** call is used (see **-video** parameter description for more details).
- the audio streams of the AVI file (that has an index order of 0 corresponding to the AVI file index) is stored in the target Moflex file with default format.
- the audio streams of the two WAV files (that have index order of 1 and 2 respectively) is stored in the target Moflex file with default format.
- output Moflex file name is video.moflex.
- no timeline is generated.

7.2.12 Use of -video parameter options for a 3D interleaved video

```
MoflexTool -avi video.avi -video stereo=interleaved rotation=none firsteye=left
```

The resulting Moflex file has the following properties:

- the first video stream of the AVI file is stored in the target Moflex file.
- video is explicitly defined as 3D interleaved left eye first and no pre-rotation performed .
- output Moflex file name is video.moflex.
- no audio.
- no timeline generated.

7.2.13 Use of -video parameter options for a 3D interleaved pre-rotated video

```
MoflexTool -avi video.avi -video stereo=interleaved rotation=90 firsteye=left
```

The resulting Moflex file has the following properties:

- the first video stream of the AVI file is stored in the target Moflex file.
- video is explicitly defined as 3D interleaved left eye first and 90° clockwise pre-rotation performed.
- output Moflex file name is video.moflex.
- no audio.
- no timeline generated.

AVI video sources should have been edited to pre-rotated by 90° clockwise before being encoded into Mobiclip format.

Pre-rotated videos can be useful if you want to lower FCRAM memory usage a bit.

7.3 MoflexTool file and stream options usage

Here is a detailed explanation of the **file** and **stream** options usage. These options can be used by the **-video**, **-audio**, and **-timeline** parameters.

Both of these options have the default value 0.

The **file** option is an index in the list of files declared with either **-avi**, **-wav** or **-moflex** parameters, starting from 0.

The **file** option is an index of the stream inside the file selected by **file** option.

If, for example, we have the following command line:

```
MoflexTool -avi video.avi -wav track0.wav -wav track1.wav ...
```

Then video.avi would have **file** option value 0, track0.wav would have **file** option value 1 and track1.wav would have **file** option value 2.

Then the first video track of the avi file would be accessed using the parameter:

```
-video
```

or

```
-video file=0
```

or

```
-video file=0 stream=0
```

The first audio track of the avi file would be accessed using the parameter:

```
-audio
```

or

```
-audio file=0
```

or

```
-audio file=0 stream=0
```

The audio track of the track0.wav file would be accessed using the parameter:

```
-audio file=1
```

or

```
-audio file=1 stream=0
```

The audio track of the track1.wav file would be accessed using the parameter:

```
-audio file=2
```

or

```
-audio file=2 stream=0
```

Revision History

Version	Revision Date	Category	Description
1.0.12	2013/04/04	-	Updated MoflexTool command line parameters to include options to info command.
1.0.11	2013/01/10	-	Updated MoflexTool command line parameters to include the type option for the timeline parameter.
1.0.10	2012/02/24	-	- Updated MoflexTool command line parameters to include the new dsp-adcpm format. - Added a note on availability of <code>WaveCodecCtr.dll</code> for encoding audio in the dsp-adpcm format.
1.0.9	2012/01/10	-	Improved drawback description of encoding schemes of Chapters 6.3.1 and 6.3.3
1.0.8	2011/10/03	-	- Removed references to direct screen access in all document (this feature is no longer supported in the Mobiclip SDK). - Minor updates in chapters 6.3.1 and 6.3.3.
1.0.7	2011/08/18	-	- Added note in Chapter 6.2.1.2 giving information about inner working of CBR encoding scheme. - Added information in Chapter 6.3.3 about VBR encoding scheme.
1.0.6	2011/07/13	-	- Added information about making video frame rate match the LCD refresh rate Chapter 3.4. - Added information on some useful Avisynth filters in Chapter 4.
1.0.5	2011/04/08	-	- Added detailed explanation of MoflexTool file and stream usage in new Chapter 6.3. - Improved description of MoflexTool -out and -info parameters. - Added 3.3 Chapter on using pre-rotated videos.
1.0.4	2011/01/07	-	AviToMoflex tool description replaced by MoflexTool description in Chapter 6.
1.0.3	2010/11/15	-	- Added description of new Quality parameter of "CBR - Nth Pass" encoding scheme in 5.2.1.6 chapter. - In Chapter 5, default values of many VFW codec parameters have been tuned to target better quality/compression ratios for all encoding schemes. - Added 5.3 section in Chapter 5 that explains usage of the different encoding schemes.
1.0.2	2010/10/08	-	- Added description of additional options of AviToMoflex tool. - Modified chapter 3 to add description of top to bottom and side by side layouts.
1.0.1	2010/09/03	-	Added description of Quality parameter in 5.2.1.1 chapter
1.0.0	2010/08/03	-	Initial version.

/

All company and product names in this document are the trademarks or registered trademarks of their respective companies.

© 2013 Nintendo

The contents of this document cannot be duplicated, copied, reprinted, transferred, distributed, or loaned in whole or in part without the prior approval of Nintendo.